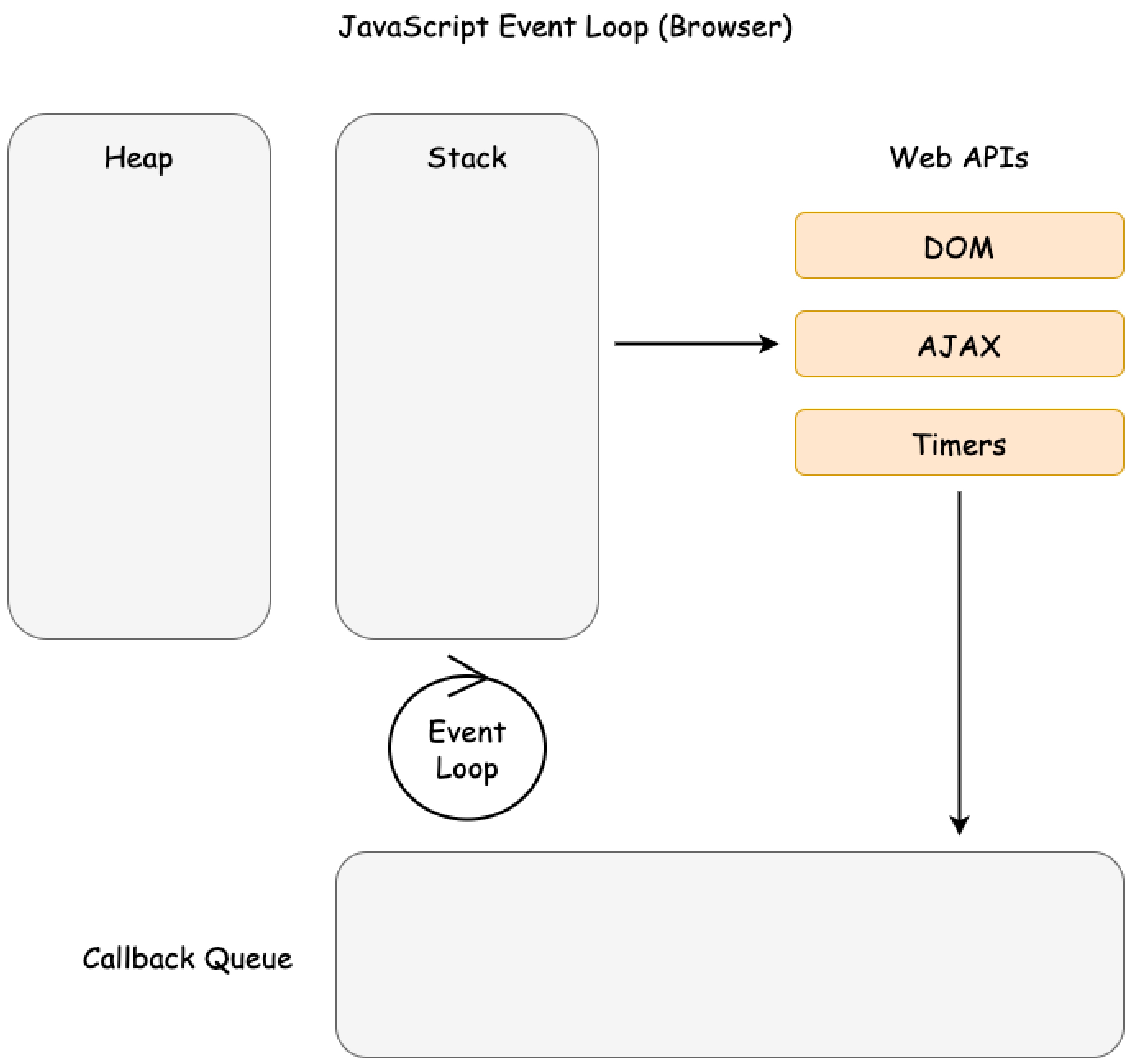
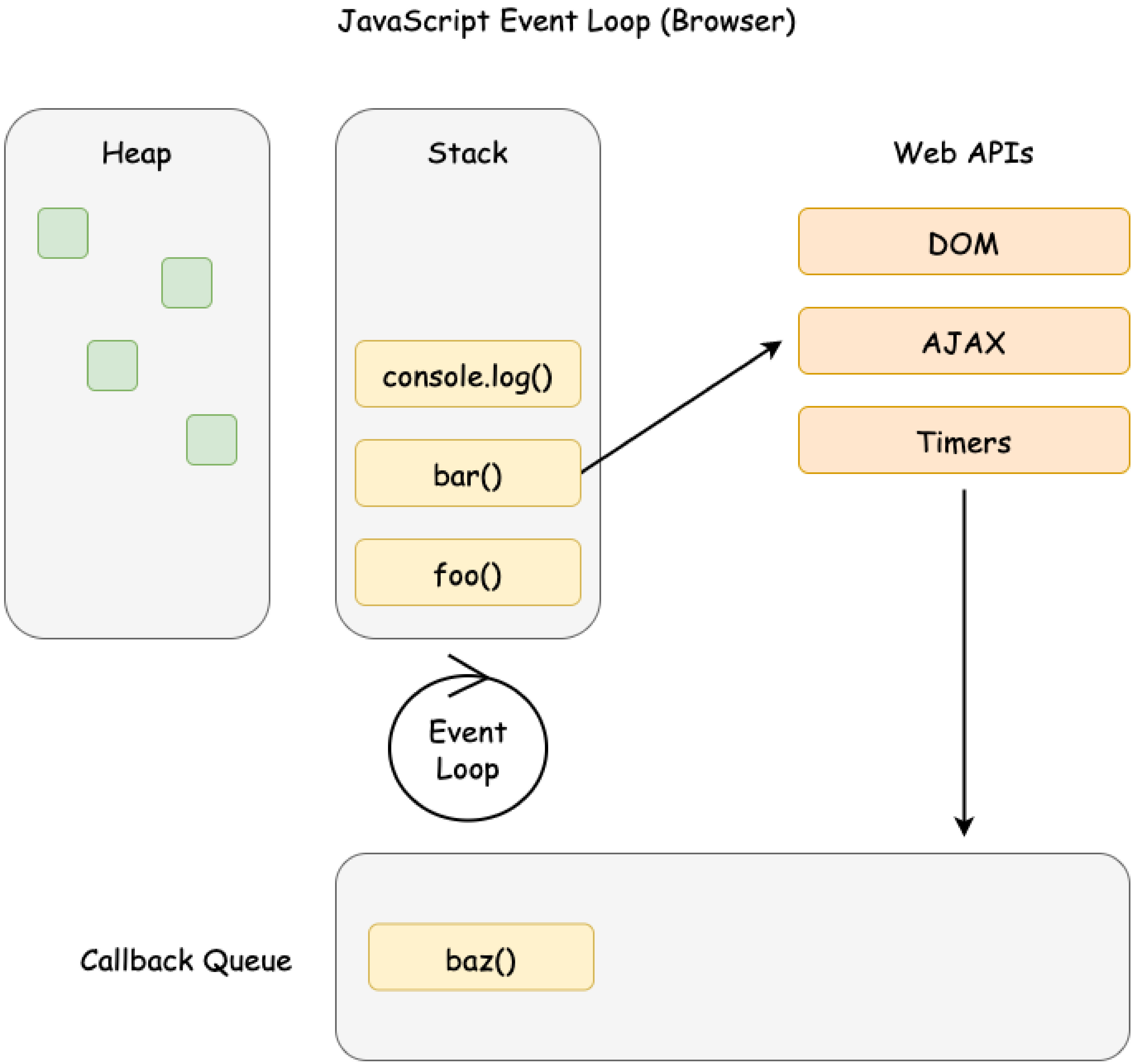
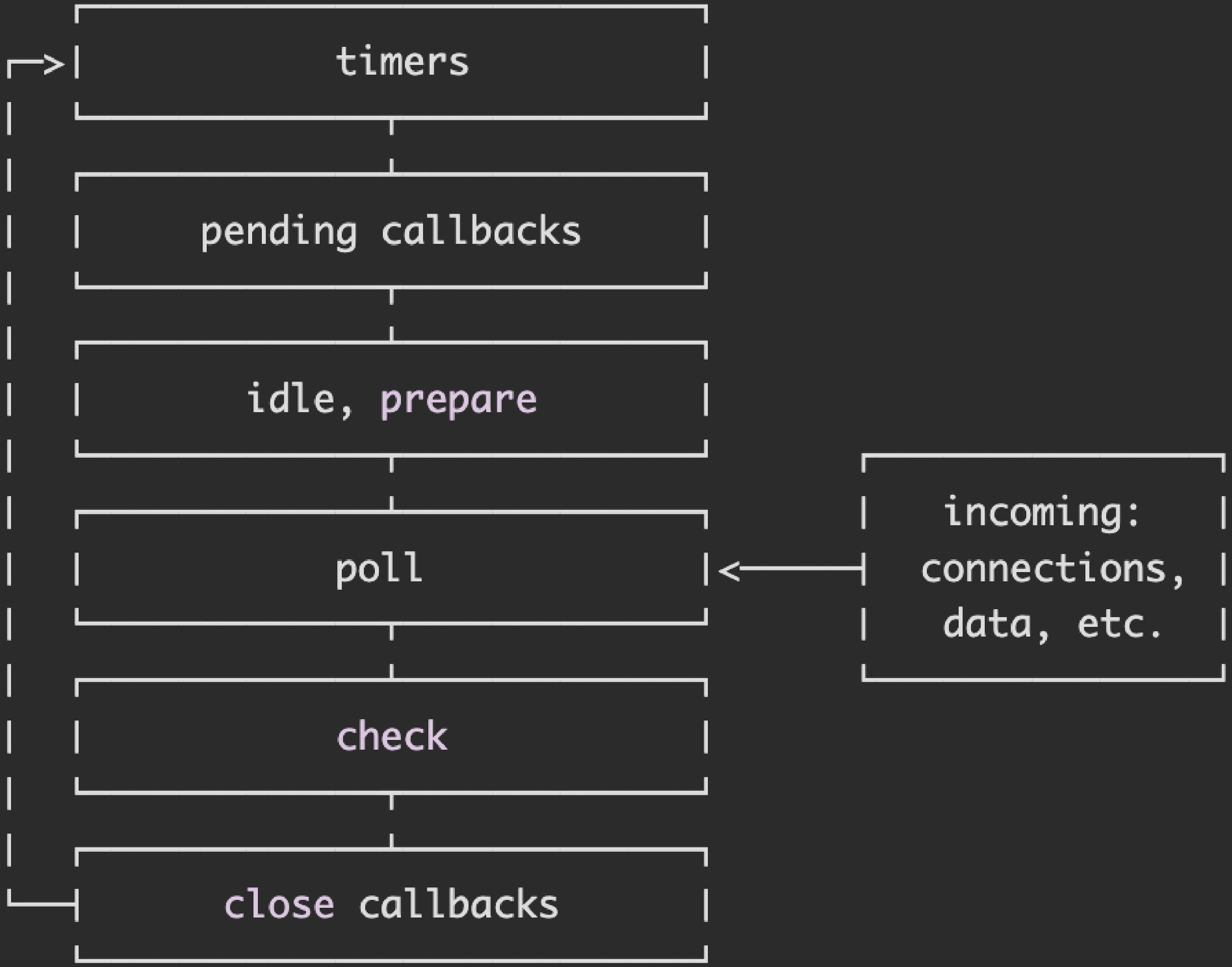
Understanding the Node.js Event Loop

https://github.com/thawkin3/nodejs-event-loop-presentation

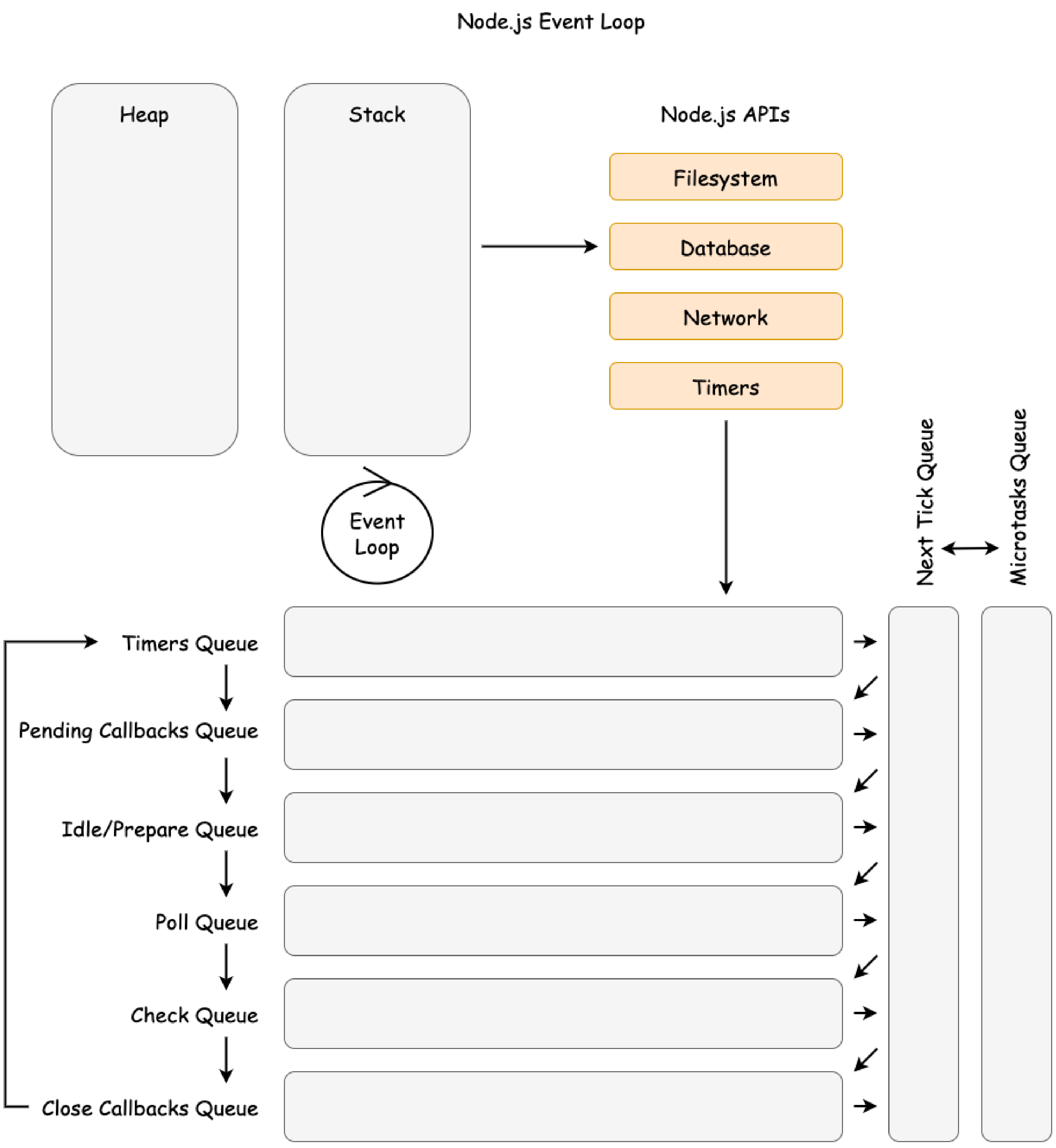

JavaScript Event Loop (Browser)
Heap
Stack
Web APIs
DOM
AJAX
Timers
Event Loop
Callback Queue


JavaScript Event Loop (Browser)
Heap
Stack
Web APIs
DOM
AJAX
Timers
console.log()
bar()
foo()
Event Loop
Callback Queue
baz()

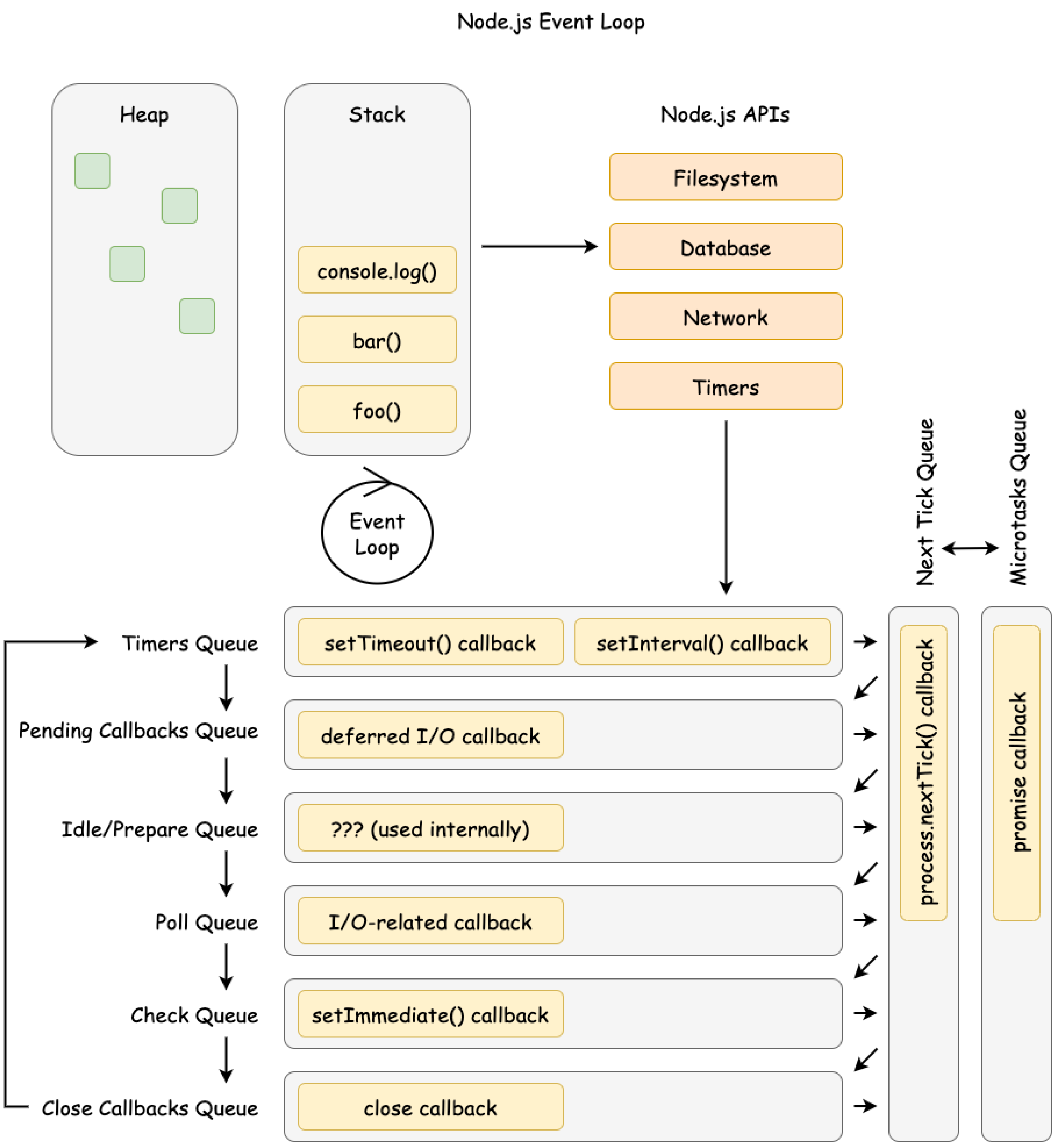

timers
pending callbacks
idle, prepare
poll
check
close callbacks
incoming:
connections,
data, etc.

Phases Overview

- **timers**: this phase executes callbacks scheduled by `setTimeout()` and `setInterval()`.
- **pending callbacks**: executes I/O callbacks deferred to the next loop iteration.
- **idle, prepare**: only used internally.
- **poll**: retrieve new I/O events; execute I/O related callbacks (almost all with the exception of close callbacks, the ones scheduled by timers, and `setImmediate()`); node will block here when appropriate.
- **check**: `setImmediate()` callbacks are invoked here.
- **close callbacks**: some close callbacks, e.g. `socket.on('close', ...)`.



Node.js Event Loop
Heap
Stack
Node.js APIs
Filesystem
Database
Network
Timers
Event Loop
Next Tick Queue
Microtasks Queue
Timers Queue
Pending Callbacks Queue
Idle/Prepare Queue
Poll Queue
Check Queue
Close Callbacks Queue


Node.js Event Loop
Heap
Stack
console.log()
bar()
foo()
Node.js APIs
Filesystem
Database
Network
Timers
Event Loop
Next Tick Queue
Microtasks Queue
Timers Queue
setTimeout() callback
setInterval() callback
Pending Callbacks Queue
deferred I/O callback
Idle/Prepare Queue
??? (used internally)
Poll Queue
I/O-related callback
Check Queue
setImmediate() callback
Close Callbacks Queue
close callback
process.nextTick() callback
promise callback

Node.js Event Loop - Main Takeaways

1. The Node.js event loop is complicated!
2. The Node.js event loop coordinates work between the call stack and the callback queues.
3. The Node.js event loop has multiple callback queues.
4. Don’t block the event loop (prefer asynchronous code over synchronous).
5. Don’t starve the event loop (with recursive calls to process.nextTick).
6. Mix and match usage of setTimeout, setImmediate, process.nextTick, and promises with care.